**DESIGN PATTERNS**

**What are Design Patterns?**

A design pattern provides a general reusable solution for the common problems that occur in software design. The pattern typically shows relationships and interactions between classes or objects. The idea is to speed up the development process by providing well-tested, proven development/design paradigms. Design patterns are programming language independent strategies for solving a common problem.

**Types of Design Patterns:-**

There are mainly three types of design patterns:-

* **Creational:** These design patterns are all about class instantiation or object creation. These patterns can be further categorized into Class-creational patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process.

**Creational Patterns are: Factory Method, Abstract Factory, Builder, Singleton, Object Pool, and Prototype**

**Use case of creational design pattern-**

1. Suppose a developer wants to create a simple DBConnection class to connect to a database and wants to access the database at multiple locations from code, generally what the developer will do is create an instance of DBConnection class and use it for doing database operations wherever required. This results in creating multiple connections from the database as each instance of DBConnection class will have a separate connection to the database. In order to deal with it, we create DBConnection class as a singleton class, so that only one instance of DBConnection is created and a single connection is established.
2. Suppose you want to create multiple instances of a similar kind and want to achieve loose coupling then you can go for Factory pattern. A class implementing factory design pattern works as a bridge between multiple classes. Consider an example of using multiple database servers like SQL Server and Oracle. If you are developing an application using SQL Server database as back end, but in the future need to change the database to the oracle, you will need to modify all your code, so as factory design patterns maintain loose coupling and easy implementation.

* **Structural:** These design patterns are about organizing different classes and objects to form larger structures and provide new functionality.

**Structural design patterns are: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Private Class Data, and Proxy.**

**Use Case Of Structural Design Pattern-**

When 2 interfaces are not compatible with each other and want to establish a relationship between them through an adapter it’s called an adapter design pattern. The adapter pattern converts the interface of a class into another interface or class that the client expects.

* **Behavioral:** Behavioral patterns are about identifying common communication patterns between objects and realizing these patterns.

**Behavioral patterns are Chain of responsibility, Command, Interpreter, Iterator, Mediator, Memento, Null Object, Observer, State, Strategy, Template method, Visitor**

**Use case Of Behavioral Design Pattern-**

Let’s you define a subscription mechanism to notify multiple objects about any events that happen to the object they're observing. This is where Observer Pattern comes into play.